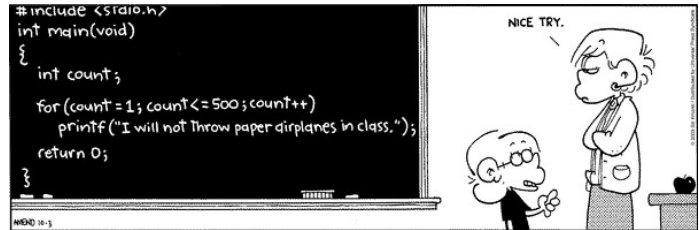


A numerical simulation project

The finite element method for solving a boundary-value problem in \mathbb{R}^2



Use C language to implement the problem.

1. Motivations

The finite element method (FEM) is now widely used to solve complex engineering problems formulated using partial differential equations. Numerical simulation based on FEM requires the development of reliable computer programs to obtain accurate solutions of the problem at hand. The objective of this semester project is to help students understand how a finite element procedure solves a problem, by writing a simple yet complete computer program.

The theoretical foundations of the finite element method and all related computational issues have been thoroughly introduced and discussed in the class lectures. However, attending the class is not sufficient to get a deep understanding on how operations combined together (weak formulation, internal approximation of the variational problem, matrix assembly, boundary conditions, linear system resolution, etc.) are effectively working. A numerical experience has been already initiated during the numerical experiments with Scilab and FreeFem++ but an excellent way of improving this experience is by writing a finite element code.

2. Problem statement

We consider a rectangular domain $\Omega =]0, L[\times]0, 1[$ representing a beam of length L that is fixed on its left boundary, denoted Γ_4 and is subjected to a volumic load $f \in L^2(\Omega)^2$ (f can be the

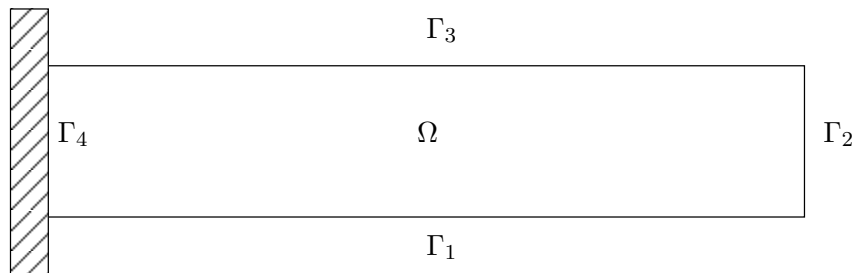


Figure 1: *The computational domain Ω for the linear elasticity problem.*

gravity, for example). The corresponding boundary-value problem is:

Find u solving:

$$\begin{cases} -\operatorname{div} \sigma(u) = f & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_4 \\ \sigma(u) \cdot n = 0 & \text{on } \partial\Omega \setminus \Gamma_4, \end{cases} \quad (1)$$

and the Hook's law gives the relation between the stress tensor σ and the linearized strain tensor e as follows:

$$\sigma = 2\mu e(u) + \lambda \operatorname{tr}(e(u)) Id, \quad \text{with} \quad e(u) = \frac{1}{2} (\nabla u + \nabla u^t),$$

λ, μ are the Lamé coefficients describing the mechanical properties of the material, that are related to the Young's modulus E and Poisson's ratio ν as follows:

$$\mu = \frac{E}{2(1+\nu)} \quad \text{and} \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}.$$

Variational formulation

In a previous numerical experiment, we have seen that if $u \in H^2(\Omega)^2$ is solution of the problem (1), it is also a solution of the following variational problem:

Find $u \in V^2$ such that:

$$\int_{\Omega} (2\mu e(u) : e(v) + \lambda(\operatorname{div} u)(\operatorname{div} v)) \, dx = \int_{\Omega} f \cdot v \, dx, \quad \forall v \in V^2, \quad (2)$$

where the Hilbert space V is the space of the functions in $H^1(\Omega)$ vanishing on Γ_4 (according to the trace theorem). Here, we suppose the body force f is the gravity.

2. A finite element program

As a semester project, each student is required to write a two-dimensional finite element code for solving the linear elasticity problem (1) using \mathbb{P}_1 Lagrange finite elements on triangular meshes. The overall organization of the project is provided as a set of C source programs. It includes various routines for reading/writing a mesh and a solution, handling sparse matrices and solving a linear system using iterative methods. The work is thus to write a generic procedure for allocating memory for the linear system, for assembling the stiffness matrix and the right-hand side vector and to call the suitable routine for solving the linear system.

- A set of meshes of different resolutions will be provided to test your finite element program.
- Special care must be taken regarding the memory resources allocation and the accuracy of the numerical solution.

4. Project schedule

Three preliminary numerical experiment classes will help the students to organize and understand the general procedure of the finite element code. The final report is due one week before the end

of the semester. A 20 minutes oral presentation of each project will take place during the last class.

The semester *project report* shall be written in \LaTeX and must include:

- a description of the mathematical problem you solved and how the weak formulation is found (in particular, how you addressed the numerical integration issues and the shape function you used),
- a short description of the program (its main features) and of the main difficulties faced,
- the possible improvements and extensions and how these would be implemented,
- several figures corresponding to numerical experiments with different parameters and variable values (for the mesh, the Lamé coefficients, the type of solver used, etc.).

Just report your coding activity on this topic, you need not review the whole program in this report.

Appendices

The details of the data structures are provided for your information; don't spend too much time going through all this information.

1. Mesh data structure

The file `elastic.h` contains all relevant data structures. The mesh geometry (*vertex coordinates*) is described in the `Point` structure:

```
typedef struct {
    double    c[3];                //    point coordinates
    int       ref;                 //    vertex reference
} Point;
typedef Point * pPoint;           //    define a pointer to structure
```

and the mesh connectivity is described in the `Tria` structure:

```
typedef struct {
    int       v[3], ref;           //    vertex connectivity and element reference
} Tria;
typedef Tria * pTria;
```

You will notice that the point structure can be used in both 2d and 3d cases. The mesh data structure is a more complex structure that contains global information about the mesh entities:

```
typedef struct {
    int       np, na, nt, ne;      //    number of mesh entities
    int       nbcl, nmat;          //    number of boundary cond. and subdomains
    int       ver, dim;           //    mesh version and dimension
    char      *name;              //    file name

    pPoint    point;              //    pointer to vertex array
```

```

pTria   tria;           //   pointer to triangle array
pTetra  tetra;
pCl     cl;             //   pointer to boundary conditions
pMat    mat;           //   pointer to subdomain properties
Info    info;          //   additional info
} Mesh;
typedef Mesh * pMesh;

```

The function `loadMesh (pMesh , char *)` is called to load mesh structure from a file and the function `saveMesh (pMesh , pSol)` allows to write the deformed mesh (after coordinates have been updated by the deformation field).

2. A sparse matrix library

As we mentioned in the class lecture, the stiffness matrix of the problem (1) is largely composed of zero elements. For obvious reasons, it is more efficient to store only non-zero coefficients in a sparse matrix structure. The Compressed Sparse Row (CSR) format is a simple and efficient format where non-zero coefficients are stored row by row in a one-dimensional array.

Structure. For a matrix $A \in \mathbb{R}^{nr,nc}$ containing nz non-zero coefficients, the CSR structure is composed of three arrays:

- `val`: a real array of size nz that contains all non-zeros elements of the sparse matrix,
- `row`: an integer array of size nr that contains the indices of the first non-zero element in each row,
- `col`: an integer array of size nz that contains the column indices of the corresponding values.

For example, suppose the matrix A is:

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 3 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 & 12 \\ 0 & 13 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{pmatrix}$$

then the contents of the three arrays will be:

`val:` 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

`col:` 1 4 6 1 2 2 3 6 1 4 5 6 2 5 6

`row:` 1 4 6 9 13 15 hence, the content of the i th row is stored in the `val` array at the successive locations:

`col[row[i]], col[row[i]+1], ..., col[row[i+1]-1]`

Routines. Various routines for creating sparse matrices and accessing sparse matrix elements are provided in the source file `sparse.c` to be compiled. In this project, the symmetric stiffness matrix is declared as an upper triangular matrix, in the function `allocA_2d`:

```
A->typ = CS_UT + CS_SYM; // (symmetric) upper triangular matrix
```

A few routines are of interest for this project:

1. `csrStoreA(pCsr A, int i, int j, double val)`: add the scalar value `val` to the coefficient a_{ij} of the sparse matrix A , i.e. $A(i, j) = val$. Notice that depending on the matrix type, the value may not be stored, if $j < i$ and matrix is upper triangular for instance;
2. `csrDiaX(pCsr A, int i, double x)`: set diagonal value $A(i, i) = x$;
3. `csrLineI(pCsr A, int i)`: set $a_{ij} = \delta_{ij}$ for line i ;
4. `csrPackA(A)`: compress the sparse matrix (optimize allocated memory);
5. `csrPrecondGrad(pCsr A, double *u, double *F, int *err, int *nit)`: solve the linear system $Au = F$ using a pre-conditioned conjugate gradient with a fixed accuracy err and a maximum number of iterations nit .

3. Parameter setting

The parameters of the linear elasticity problem can be specified in a file `DEFAULT.elas` associated with the mesh. The structure of the file is the following:

```
# boundary conditions // comments start with the # sign
Dirichlet // type of boundary condition
2 // number of boundary condition
1 Vertices V 0. 0. // condition applied on all vertices of ref 1
2 Vertices V 0. -0.2

# body force
Gravity // keyword
0. -10.

# Coefficients
Lame // keyword for Lamé coefficients (lambda mu)
1 // number of materials on which the coefficients are applied
0 186000. 3400. // reference of subdomain and numerical values of Lamé coeffs.
```

4. Visualizing the results

The numerical solution obtained by your finite element program can be visualized using `medit` free-software visualization code that will be provided. By default, the solution is saved in a file `file.sol` associated with the mesh file `file.mesh`. Color Postscript figures of the solution and of the related mesh deformation can be easily saved using `medit` to illustrate your report.